National Institute of
Standards and Technology
U.S. Department of Commerce

NIST Interagency Report 7697
(DRAFT)

Common Platform Enumeration: Dictionary Specification Version 2.3 (DRAFT)

Paul Cichonski
David Waltermire

8

9 **NIST Interagency Report 7697 (DRAFT)**

Common Platform Enumeration: Dictionary Specification Version 2.3 (DRAFT)

Paul Cichonski
David Waltermire

C O M P U T E R S E C U R I T Y

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

August 2010

U.S. Department of Commerce

Gary Locke, Secretary

National Institute of Standards and Technology

Dr. Patrick D. Gallagher, Director

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analysis to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Interagency Report discusses ITL's research, guidance, and outreach efforts in computer security and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Interagency Report 7697 (DRAFT)
61 pages (August 2010)

Acknowledgments

The authors, Paul Cichonski of Booz Allen Hamilton and David Waltermire of NIST wish to thank their colleagues who reviewed drafts of this document and contributed to its technical content. The authors would like to acknowledge Harold Booth of NIST, Chris McCormick of Booz Allen Hamilton, Seth Hanford of Cisco Systems, Inc., Tim Keanini of nCircle, Kent Landfield of McAfee, Inc., Brant A. Cheikes and Mary Parmelee of the MITRE Corporation, Jim Ronayne of Cobham plc, and Shane Shaffer of G2, Inc. for their insights and support throughout the development of the document.

Abstract

This report defines the concept of a Common Platform Enumeration (CPE) Dictionary, the rules associated with CPE Dictionary creation and management, and the data model for representing a CPE Dictionary. The CPE Dictionary Specification is a part of a stack of CPE specifications that serves to support a variety of use cases relating to IT product description and naming. An individual CPE Dictionary is a repository of IT product names, with each name in the repository identifying a unique class of IT product in the world. This specification defines the semantics of the CPE Dictionary data model. This common semantics provides a shared understanding, of CPE Dictionary constructs, to users from different communities of practice. This specification also defines the methodology for capturing official IT product names through an Official CPE Dictionary construct.

Audience

This specification document defines standardized data models and machine encodings for creating product descriptions and identifiers. These models and encodings are envisaged to be of interest to the following audiences:

a. **Asset inventory tool developers.** Asset inventory tools inspect computing devices and assemble catalogs that list installed component hardware and software elements. In the absence of CPE, there is no standardized method for how these tools should report what they find. The CPE specification stack provides all the technical elements needed to comprise such a capability. Furthermore, CPE is intended to address the needs of asset inventory tool developers regardless of whether the tools have credentialed (authenticated) access to the computing devices subject to inventory.

b. **Security content automation tool developers.** Many security content automation tools are fundamentally concerned with making fully- or partially-automated information system security decisions based on collected information about installed products. The CPE specification stack provides a framework that supports correlation of information about identical products installed across the enterprise, and association of vulnerability, configuration, remediation and other security-policy information with information about installed products.

c. **Security content authors.** Security content authors are concerned with creating machine-interpretable documents that define organizational policies and procedures pertaining to information systems security, management and enforcement. Often there is a need to tag guidance, policy, etc., documents with information about the product(s) to which the guidance, policy, etc., applies. These tags are called *applicability statements*. The CPE specification stack provides a standardized mechanism for creating applicability statements which can be used to ensure that guidance is invoked as needed when the product(s) to which it applies is discovered to be installed within an enterprise.

Table of Contents

List of Figures

List of Tables

Executive Summary

Following security best practices is essential to maintaining the security and integrity of today's Information Technology (IT) systems and the data they store. Given the speed with which attackers discover and exploit new vulnerabilities, best practices need to be continuously refined and updated as fast as or faster than the attackers can operate. To this end, *security automation* has emerged as an advanced computer-security technology intended to help information system administrators assess, manage and maintain the security posture of their IT infrastructures regardless of their enterprises' scale, organization and structure. The United States government, under the auspices of the National Institute of Standards and Technology (NIST), has established the Security Content Automation Protocol (SCAP) to foster the development and adoption of security automation standards and data resources.[1]

The *Common Platform Enumeration* (CPE) addresses the security automation community's need for a standardized method to identify and describe the software systems and hardware devices present in an enterprise's computing asset inventory. Four specification documents comprise the CPE stack:

1. Naming
2. Matching
3. Dictionary
4. Language

The Naming specification defines the logical structure of well-formed CPE names (WFNs), and the procedures for binding and unbinding WFNs to and from machine-readable encodings. The Name Matching specification defines the procedures for comparing WFNs to determine whether they refer to some or all of the same products or platforms. The Dictionary specification—this document—defines the concept of a dictionary of identifier WFNs, and prescribes high-level rules for dictionary curators. The Language specification defines a standardized structure for forming complex logical expressions out of WFNs. These four specifications are arranged in a *specification stack* as depicted in Figure ES-1.

Figure ES-1: CPE Specification Stack

Collectively, the CPE Specification Stack aims to deliver these capabilities to the security automation community:

- A method for assigning unique machine-readable identifiers to certain classes of IT products and computing platforms;
- A method for curating (compiling and maintaining) dictionaries (repositories) of machine-readable product and platform identifiers;
- A method for constructing machine-readable referring expressions which can be mechanically compared (i.e., by a computer algorithm or procedure) to product/platform identifiers to determine whether the identifiers satisfy the expressions;

[1] For more information on SCAP, cf. NIST Special Publication 800-117, *Guide to Adopting and Using the Security Content Automation Protocol*, http://csrc.nist.gov/publications/drafts/800-117/draft-sp800-117.pdf.

- A set of interoperability requirements which guarantee that heterogeneous security automation tools can select and use the same unique identifiers to refer to the associated products and platforms.

1. Introduction

1.1 Purpose and Scope

This document defines the policies associated with creating a Common Platform Enumeration (CPE) Dictionary. A CPE dictionary is a repository of CPE Names that identify an individual product; the dictionary stores both the enumeration of these CPE Names as well as metadata associated with the CPE Names. This document formally defines the concept of a CPE dictionary, the formal rules relating to dictionary instantiation and management, and the data model that represents all dictionary concepts and relationships. This document also establishes the concept of an Official CPE Dictionary, as well as the process for how organization can extend the Official CPE Dictionary using Extended CPE Dictionaries.

A CPE dictionary is a repository of CPE Names, where each name in the dictionary identifies a single class of IT product in the world. The word 'class' here signifies that the object identified is not a physical instantiation of a product on a system, but rather the abstract model of that product. In addition to using a CPE name to identify a single product class, organizations may also use CPE names to represent a set of multiple product classes. This is a very important distinction, and one must understand that a CPE dictionary stores only CPE Names that identify a single product class, not a set of product classes.

The scope of this document is limited to formally defining the CPE dictionary concept as well as rules associated with dictionary instantiation and management. This document does not include normative guidance relating to other components of the CPE stack such as CPE Names, CPE matching algorithms, or CPE language structure.

1.2 Normative References

The following documents are indispensible references for understanding the application of this specification.

[CPE22] Buttner, A. and N. Ziring. (2009). *Common Platform Enumeration—Specification*. Version 2.2 dated 11 March 2009. See: http://cpe.mitre.org/specification/spec_archive.html.

[CPE23-N] Cheikes, B. A. and Waltermire, D. (2010). *Common Platform Enumeration: Naming Specification Version 2.3.*

[CPE23-M] Parmelee, M. C., Booth, H. and Waltermire, D. (2010). *Common Platform Enumeration: Name Matching Specification Version 2.3*

[ISO19770-2] ISO/IEC 19770-2. (2009). *Software Identification Tag.* November 2009. See: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=53670.

[SCAP800-117] NIST Special Publication 800-117, *Guide to Adopting and Using the Security Content Automation Protocol.* See: http://csrc.nist.gov/publications/drafts/800-117/draft-sp800-117.pdf.

[TUCA] *Common Platform Enumeration (CPE) Technical Use Case Analysis.* White Paper, The MITRE Corporation, November 2008. See: http://cpe.mitre.org/about/use_cases.html.

1.3 Document Structure

The remainder of this document is organized into the following major sections:

- Section 2 defines the terms used within this specification, and provides a list of common abbreviations.
- Section 3 defines the conformance rules for this specification.
- Section 4 provides an overview of related specifications or standards.
- Section 5 defines the high-level rules and acceptance criteria relating to a CPE dictionary.
- Section 6 defines the valid searching operations relating to a CPE Dictionary.
- Section 7 defines the required management documents relating to a CPE dictionary.
- Section 8 provides an overview of the Official and Extended CPE Dictionary concepts.
- Section 9 defines the CPE dictionary data model.
- Section 10 provides pseudo code that implements the various concepts defined in other sections of the specification. This section also defines the requirements for valid bindings of the CPE dictionary data model.

The document also contains appendices with informative reference material.

- Appendix A provides potential use cases relating to this specification.
- Appendix B provides an example of the disparate Identification Strategies for IT products.
- Appendix C provides a listing of valid bindings for the CPE dictionary data model.
- Appendix D provides a change-log that documents the changes made during public review period.

1.4 Document Conventions

This specification adheres to all rules and conventions defined lower in the CPE stack of specifications. The CPE Naming Specification [CPE23-N]defines the concept of a Well-Formed CPE Name (WFN) that is a logical representation of a CPE name. Wherever possible, this specification uses WFN representation of CPE names to limit the dependency on any concrete form of CPE name binding.

The CPE Naming Specification [CPE23-N:1.2.1] defines two primary uses of a WFN. The first use case for a WFN is to describe a set of product classes in existence. The second use case for a WFN is to define a single, unique, product class in existence. A CPE Dictionary is a collection of WFNs supporting the second use case, in other words all non-deprecated WFNs within a CPE dictionary uniquely identify a single product class. This specification will always use of the term *identifier WFN* to represent a WFN that is uniquely identifying a single product class.

Text intended to represent computing system input, output, or algorithmic processing is presented in `fixed-width Courier font`.

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in Request for Comment (RFC) 2119.[2]

[2] RFC 2119, “Key words for use in RFCs to Indicate Requirement Levels”, is available at http://www.ietf.org/rfc/rfc2119.txt.

2. Terms, Definitions and Abbreviations

2.1 Terms and Definitions

This section defines a set of common terms used within the document. This section builds on the terms defined in CPE Naming Specification [CPE23-N] and CPE Matching specification [CPE23-M] and does not repeat them here.

2.1.1 Dictionary Contributor

Any organization or person that submits new CPE Names to the dictionary for inclusion.

2.1.2 Dictionary Creator

A *dictionary creator* is any organization that instantiates a CPE dictionary that conforms to the guidance within this specification.

2.1.3 Dictionary Maintainer

A *Dictionary Maintainer* is any organization that manages a CPE dictionary and all processes relating to that CPE dictionary. In the majority of cases, the organization that serves as the dictionary creator will also serve as the *Dictionary Maintainer* for a specific CPE dictionary.

2.1.4 Dictionary Users

Any organization, individual, or software product that consumes a CPE dictionary for any purpose.

2.1.5 CPE Specification Stack

The CPE Specification Stack is the set of all specifications that make up CPE 2.3. At the time of writing this specification, the set includes The Common Platform Enumeration: Naming Specification Version 2.3, The Common Platform Enumeration: Name Matching Specification Version 2.3, The Common Platform Enumeration: Dictionary Specification Version 2.3, and The Common Platform Enumeration: Language Specification Version 2.3

2.1.6 Extended CPE Dictionary

An *Extended CPE dictionary* is a dictionary that an organization may stand up to house *identifier WFNs* not found in the *Official CPE Dictionary*. Section 8.2 defines this concept fully.

2.1.7 Identification Strategy

An *identification strategy* is the way in which a particular product is versioned, or named. More specifically this includes the way the product is versioned, the syntax of a version string for a product, and the semantics implied by that version string. Appendix B— expands upon this definition in detail.

2.1.8 Identifier WFN

A logical representation of a CPE that is uniquely identifying a single product class.

2.1.9 Known Data

Known Data is a term used to describe one category of information that may be present within an attribute of a CPE Name. *Known Data* represents any meaningful value about a product (e.g. sp1, 2.3.4, pro, NA), but does not include values such as ANY.

2.1.10 Official CPE Dictionary

The *Official CPE Dictionary is* the authoritative repository of identifier WFNs hosted by NIST. Section 8.1 defines this concept fully.

2.1.11 Official Identifier WFN

Any logical representation of a CPE that uniquely identifies a single product class and is contained within the *Official CPE Dictionary.*

2.2 Abbreviated Terms

CPE	Common Platform Enumeration
IT	Information Technology
NIST	National Institute of Standards and Technology
SCAP	Security Content Automation Protocol
WFN	Well-formed CPE Name
URI	Uniform Resource Identifier

3. Conformance

Products and organizations may want to claim conformance with this specification for a variety of reasons. For example, a product may want to assert that it uses official identifier WFNs internally and can interoperate with any other product using official identifier WFNs. Another example may be that a policy mandates that organizations use CPE to track and identify products on their systems.

This section provides the high-level requirements that a product or organization must meet if they are seeking conformance with this specification. The majority of the requirements listed in this section reference other sections in this document that fully define the high-level requirement.

3.1 Product Conformance

This section contains the set of requirements for IT products asserting conformance with The CPE 2.3 Dictionary Specification. Products may claim conformance with this specification to show that the product uses identifier WFNs internally and can interoperate with other products using identifier WFNs. All products claiming conformance with this specification MUST adhere to the following requirements:

1. Products using identifier WFNs SHOULD only use official identifier WFNs that are located in the Official CPE Dictionary; if an official identifier WFN is not available, the tool MUST use an identifier WFN contained within an Extended CPE dictionary to which it has access.

2. When a product consumes or outputs an identifier WFN, then that product MUST first determine if the identifier WFN is deprecated in the Official CPE Dictionary. If the identifier WFN is not present in the Official CPE Dictionary, then the product MUST determine if the identifier WFN is deprecated in an Extended CPE Dictionary to which it has access. If the identifier WFN is deprecated, the product MUST resolve the correct non-deprecated identifier WFNs for use in place of the deprecated identifier WFN. The product MUST follow the process specified in section 5.2.3 to perform this resolution. This requirement MAY be ignored if the product is intending to communicate information about deprecated identifier WFNs.

3.2 Organization Conformance

Organizations creating or maintaining CPE dictionaries may claim conformance with the CPE 2.3 Dictionary Specification if that organization meets the requirements in this section. This section first provides the generic requirements for all organizations claiming conformance with the CPE 2.3 Dictionary Specification. This section then provides the requirements specific to the type of dictionary the organization is creating/maintaining (i.e. Official CPE Dictionary vs. Extended CPE dictionary).

1. Organizations creating/maintaining a CPE dictionary MUST adhere to the minimum set of identifier WFN acceptance criteria defined in section 5.1. Organizations MAY further restrict these acceptance criteria, but organizations MUST NOT conflict with the criteria.

2. Organizations creating/maintaining a CPE dictionary MUST adhere to the identifier WFN deprecation process defined in section 5.2.2 when performing deprecation.

3. Organizations creating/maintaining a CPE dictionary MUST capture the identifier WFN provenance data defined in section 5.3.

4. Organizations creating/maintaining a CPE dictionary MUST ensure that the data captured within their dictionary adheres to the data model defined in section 9; this includes ensuring that all entries in the dictionary contain the mandatory elements defined in section 9.1.

5. Organizations creating/maintaining a CPE dictionary MAY extend the CPE dictionary data model defined in section 9. These organizations MUST use only the extension points defined in section 9.3 of this specification when extending the CPE dictionary data model.

6. Organizations creating/maintaining a CPE dictionary MUST use a dictionary binding that is valid according to the requirements specified in section 10.

7. Organizations creating/maintaining a CPE dictionary SHOULD create and maintain the CPE Management Documents defined in section 7. If an organization does not create organic CPE Management Documents, then the organization MUST adhere to a known set of CPE Management Documents from a different dictionary.

8. If an organization is maintaining the Official CPE Dictionary, then that organization MUST adhere to the requirements for the Official CPE Dictionary defined in section 8.1.

9. If an organization is creating an Extended CPE dictionary, then that organization MUST adhere to the requirements for Extended CPE dictionaries defined in section 8.2.

4. Relationship to Existing Standards

This section is primarily informative, and is intended to define the relationship between this specification and any related specifications or standards (both current and past).

4.1 Relationship to CPE Specification Stack

As mentioned earlier, this specification is the foundation of the CPE Specification Stack, a suite of specifications that together comprise Version 2.3 of the CPE *specification stack*. This document builds upon The CPE Naming Specification [CPE23-N] and The CPE Name Matching Specification [CPE23-M] to define the concept of a CPE dictionary. This document uses the concepts defined in the naming and matching specifications to formalize the idea of a repository of product identifiers and associated metadata.

4.2 Relationship to CPE v2.2

The CPE Specification Stack, including this specification, is intended to replace [CPE22]. Whereas [CPE22] defined all elements of CPE in a single document, starting with this release we have changed the organization of CPE to use a stack-based model. In the stack model, capabilities are built incrementally out of simpler, more narrowly defined elements that are specified lower in the stack. This design opens opportunities for innovation, as novel capabilities can be defined by combining only the needed elements, and the impacts of change can be better compartmentalized and managed. The CPE Specification Stack is intended to provide all the capabilities made available by [CPE22] while adding new features suggested by the CPE user community.

5. Rules and Acceptance Criteria

This section defines the high-level rules and acceptance criteria relating to identifier WFNs contained within a CPE dictionary.

5.1 Acceptance Criteria

If an organization is a Dictionary Maintainer, then that organization MUST only permit valid identifier WFNs in the dictionary[3]; an identifier WFN is valid if it meets the acceptance criteria defined in the following sub-sections.

The purpose of the acceptance criteria defined in the following sub-sections is to ensure a certain degree of interoperability across all CPE dictionaries in existence. The CPE Dictionary Specification achieves this interoperability through the standardized restriction of identifier WFNs permitted in a CPE dictionary. This standardized restriction ensures that dictionary users may rely on a specific level of format and quality relating to identifier WFNs stored in all CPE dictionaries in existence.

As mentioned previously, a CPE dictionary is a collection of identifier WFNs that serve to identify classes of products that exist in the world. To support this idea, the acceptance criteria focus on ensuring that each identifier WFN contained within a CPE dictionary identifies a single product class that exists in the world. These acceptance criteria represent the minimum set of rules required for WFNs within a dictionary. Dictionary Maintainers MAY further restrict the acceptance criteria defined in this section, but the maintainers MUST NOT relax the criteria. The Official Dictionary is permitted to further restrict these rules as well, but SHOULD NOT since it will require all Extended Dictionaries to implement the restrictions.

5.1.1 Permitted Concepts and Special Characters

CPE Naming Specification [CPE23-N:5.5.1] reserves a set of logical values and special characters for use within a CPE attribute value. The CPE Dictionary Specification permits a subset of these logical values to be contained within dictionary identifier WFNs.

5.1.1.1 NA Logical Value

Identifier WFNs contained within a CPE dictionary MAY contain the NA logical value within CPE attribute values, but CPE Names MUST NOT include the NA logical value within part, vendor or product attributes.

The NA value within a CPE attribute signifies that the attribute is Not Applicable for a specific product. For example, a vendor may distribute the first release of a product that contains no update, but later releases an update to this first release in the form of a service pack. This example contains two distinct products, the first product that contains no update, and the second product that does contain an update. The identifier WFN created to represent the first release of the product would contain an update attribute with a value of NA (i.e. update="NA") since the update attribute is known to be not applicable to the product. The identifier WFN created to represent the second release of the product would contain an update attribute with a value of sp1 (i.e. update="sp1").

[3] Pseudo code that implements this rule is found in section 10.1.2, in the function `accept-name(w, d)`.

5.1.1.2 ANY Logical Value

Identifier WFNs contained within a CPE dictionary MAY contain the ANY logical value within CPE attribute values, but identifier WFNs MUST NOT include the ANY logical value within part, vendor, product, or version attributes[4].

When a Dictionary Contributor uses ANY within a CPE attribute, it signifies that there is not enough information to populate the specific CPE attribute at the time of name creation. The need for this logical value originates from the mixing of semantics within the identifier WFN. It is normal that over any given time span the amount of information known about a product will grow. To support this growth of information within a CPE it is necessary to evolve the identifier WFN itself. For example, when an organization discovers more information about a given product represented by an existing identifier WFN in the dictionary they must modify that identifier WFN to include the new information. Section 5.2.2 of this specification defines the formal deprecation process for tracking changes to identifier WFNs within a dictionary.

Showing the full evolution of an identifier WFN requires showing the start of the process, which in some cases is an identifier WFN with a certain amount of missing data. To show changes to this identifier a CPE dictionary MUST capture the missing data using the ANY logical value within the specific CPE attribute for which data is missing. It is necessary to explicitly represent this concept within an identifier WFN to allow the identifier WFN to formally evolve, as more information becomes known relative to the product identified.

5.1.2 Restricted Concepts and Special Characters

CPE Naming Specification [CPE23-N:5.5.2] reserves a set of special characters for use within a CPE attribute value. The CPE Matching Specification [CPE23-M:5.1.3] assigns specific meaning to a subset of these special characters. The matching specification defines these characters as wild card characters. This wild-card concept works well when creating expressions to represent a set of products, but it does not add value to a WFN attempting to identify a single product. Therefore, the meaning imposed on these characters invalidates their use within the attribute values of an identifier WFN. CPE Dictionaries MUST NOT contain any identifier WFNs with attribute values containing any of the values specified in the following subsections, however CPE Dictionaries MAY contain these values within WFNs not serving as identifiers (e.g. within deprecation logic). CPE names containing these values are not valid identifier WFNs[5].

5.1.2.1 Asterisk

Identifier WFNs contained within a CPE dictionary MUST NOT contain the Asterisk character within any attribute of the CPE Name, unless the Asterisk is escaped. The CPE Matching Specification [CPE23-M:5.1.3] defines the Asterisk as a multi-character, embeddable wildcard. The concept of an embedded wildcard character works well when creating expressions to describe a set of products, but it does add not value to a WFN attempting to identify one, and only one, product.

[4] Section 4.2.3 provides a complete description of the reasoning for disallowing the ANY value in the part, vendor, product, version attributes.

[5] Pseudo code that implements this rule is found in section 10.1.2, in the function `contains-restricted-characters(w).`

5.1.2.2 Question-mark

Identifier WFNs contained within a CPE dictionary MUST NOT contain the Question-mark character within any attribute of the CPE Name, unless the question-mark is escaped. The CPE Matching Specification [CPE23-M:5.1.3] defines the question-mark as a single-character, embeddable wildcard. The concept of an embedded wildcard character works well when creating expressions to describe a set of products, but it does not add value to a WFN attempting to identify one, and only one, product.

5.1.3 CPE Name Completeness - Required CPE Attributes

Identifier WFNs contained within a CPE dictionary MUST contain known data for the part, vendor, product, version attributes and this data MUST NOT include the ANY logical value. Also, identifier WFNs within a CPE dictionary MUST NOT include the NA logical value in the part, vendor or product attributes, but the NA value MAY be used in the version attribute. This rule is in place to ensure that all identifier WFNs within a CPE dictionary contain the minimum amount of data required to identify a unique product class.

This requirement enforces a minimum degree of completeness for the identifier WFNs within a CPE dictionary. This specification determines identifier WFN completeness by analyzing the amount of known data within the attributes of an identifier WFN. The ANY logical value is useful for capturing the fact that data for certain attributes of an identifier WFN has not yet been discovered. However, ANY does not increase the completeness of an identifier WFN, only known data can increase the identifier WFN completeness. Known data here refers to any value that represents some distinct aspect of a product (e.g. “sp1”, “2.0”, “Microsoft”, “ios”), this includes the NA logical value since it represents the known fact that absolutely no data exists for a specific CPE attribute.

5.1.4 CPE Name Uniqueness

Every identifier WFN contained within a CPE dictionary MUST be unique in the sense that it MUST NOT result in a superset match[6] against a more complete identifier WFN contained within the dictionary. The term 'match' means matching as specified in The CPE Matching Specification [CPE23-M], and more specifically in section 6.2 of this specification. This requirement exists to enforce the fact that a CPE dictionary is a collection of product identifiers, where each CPE Name within the dictionary represents one unique class of product in the world.

Dictionary Maintainers must use matching to determine CPE uniqueness to deal with the fact that CPE Names containing the ANY logical value may exist within the dictionary[7]. As discussed in section 5.1.1.2, it is possible for CPE Name creators to submit names to the dictionary to identify products without having the full set of information required to populate all CPE attributes with known data. CPE dictionaries must support this situation since lack of information is a reality in the IT world. At the same time, dictionaries MUST NOT include these non-complete names if a more complete version of that name exists in the dictionary.

To provide an example (focusing on the CPE update attribute), suppose that a CPE dictionary contains the following identifier WFN version of a CPE:

[6] The concept of a "superset match" is formally defined in the CPE Matching Specification [CPE23-M:6.2]

[7] Pseudo code that implements this rule is found in section 10.1.2, in the function `matches-more-complete-in-dictionary(w, d)`.

```
wfn:[part="a", vendor="foo_company", product="bar", version="2.3"
update="sp1", edition="ANY", language=="ANY", sw_edition="ANY",
target_sw="ANY", target_hw="ANY", other=="ANY"]
```

The above CPE represents the hypothetical product: Foo Company Bar 2.3 sp1. At some later point in time, a Dictionary Contributor submits the following identifier WFN version of a CPE for potential inclusion within the same dictionary:

```
wfn:[part="a", vendor="foo_company", product="bar", version="2.3"
update="ANY", edition="ANY", language="ANY", sw_edition="ANY",
target_sw="ANY", target_hw="ANY", other="ANY"]
```

This new WFN represents the hypothetical product: Foo Company Bar 2.3. The difference with this new WFN is that "ANY" is the value for the update attribute. This means that the contributor of this WFN does not know the correct value of the update field at the time of submission. The acceptance criteria defined in this document does not permit this submission because an identifier WFN already exist that contains a known values for the update field of the Bar 2.3 product. The reason for this restriction is that the less-complete WFN does not represent any real-world product, but instead represents a set of existing identifier WFNs within the dictionary. If the Dictionary Contributor discovers a new update of the product Foo Company Bar 2.3, then the contributor should submit an identifier WFN containing the new update value.

This does not mean that an identifier WFN representing the product Foo Company Bar 2.3 with no update is not permitted (e.g., the first release of the product contained no update). The identifier WFN representing the product with no update would contain the NA logical value in the update attribute. The following CPE does meet the acceptance criteria since it represents a real-world product that is known to have no update (i.e. the initial release of Bar 2.3):

```
wfn:[ part="a", vendor="foo_company", product="bar", version="2.3"
update="NA", edition="ANY", language="ANY", sw_edition="ANY",
target_sw="ANY", target_hw="ANY", other="ANY"]
```

5.2 CPE Dictionary Deprecation Process

CPE Deprecation is a process specific to CPE dictionaries. When a Dictionary Maintainer deprecates an identifier WFN within a dictionary, it signifies that the deprecated identifier WFN is no longer valid. This means that CPE dictionary users MUST NOT use deprecated identifier WFNs.

The process of CPE Deprecation is necessary to support the fact that all identifier WFNs stored within the CPE dictionary MUST be immutable. This requirement for immutability derives from the fact that the data relating to the product identified is captured in the identifier itself. This means that the Dictionary Maintainer must modify the identifier WFN to modify the data about the product. The problem here evolves from the fact that as soon as the Dictionary Maintainer modifies the identifier WFN any historical information about that identifier WFN disappears, including evidence of the identifier's existence. This disappearance will cause problems for any CPE User already using the old identifier WFN. Restricting the ability to change identifier WFNs within a CPE dictionary resolves this problem.

The fact that identifier WFNs within a CPE dictionary are immutable means that any updates to the product data captured in the identifier WFN must occur through deprecation. For example, when an organization discovers more information about a product represented by an existing identifier in the dictionary they must modify that identifier to include the new information. To support this modification

of an identifier WFN it is necessary to deprecate the legacy identifier WFN in favor of the new identifier WFN, or set of identifier WFNs.

An important distinction exists between the set of names that deprecate an identifier WFN and the identifier WFN that a CPE User selects as a replacement of the deprecated name. The CPE dictionary may link a deprecated identifier WFN to a set of identifier WFNs replacing it, but this is done for informational purposes. When making this deprecation relationship the Dictionary Maintainer is asserting any identifier WFN within the set of new identifier WFNs is a valid replacement for the deprecated identifier WFN. Using this information a CPE dictionary user may, depending on the use case, decide on the appropriate name to use as a replacement, or decide to use the entire set. For example, when resolving the deprecating entries for deprecated name found within an applicability statement for a vulnerability an organization may find it useful to use the entire set to avoid possible false negatives.

Section 5.2.1 defines the different types of deprecation that may occur within a CPE dictionary. Section 5.2.2 describes how this deprecation process works in the context of a CPE dictionary. Section 5.2.3 describes how CPE dictionary users should process deprecated identifier WFNs. Section 9 describes the data model for capturing all information relating to the deprecation process.

5.2.1 Deprecation Types

This section defines three distinct types of deprecation that a Dictionary Maintainer may perform; these types include Identifier WFN Correction, Identifier WFN Removal, and Additional Information Discovery. When performing CPE deprecation, the Dictionary Maintainer MUST choose the type of deprecation they are performing. The following list defines the three types in detail:

- **Identifier WFN Correction** – An error occurred during name creation and the Dictionary Maintainer must update the identifier WFN to fix this error. The specific type of error may vary between a range of possibilities including a syntax error (e.g. typo, misspelling), or an incorrect product listing. For example, a Dictionary Maintainer may add a product to the dictionary, only to later discover that the added product is only a component library of a larger product. In this case, the dictionary maintainer would deprecate the identifier WFN representing the component library to the identifier WFN representing the actual product containing the component. This type of deprecation is *one-to-one*, with the deprecated WFN pointing to the single new identifier WFN that represents the correct product.

- **Identifier WFN Removal** – An identifier WFN exists in the dictionary that does not belong and has no replacement. This condition normally results from a case where the Dictionary Maintainer makes an error that cannot be corrected with a new identifier WFN (i.e. the name should never have been included in the dictionary). In this case, Dictionary Maintainer will deprecate the legacy name without pointing to a new name.

- **Additional Information Discovery** – The Dictionary Maintainer adds one or more identifier WFNs to the dictionary that are more complete than an existing identifier WFN within the dictionary. In this case, the pre-existing name really represents a set of possible products that the maintainer did not know about when originally adding the name to the dictionary. This type of deprecation may be *one-to-many* since the Dictionary Maintainer will deprecate the pre-existing name to all of the names that are more complete. This deprecation relationship is largely informational, when making this deprecation the Dictionary Maintainer is asserting that any identifier WFN name within the set of new

identifier WFNs is a valid replacement for the deprecated WFN; it is up to the Dictionary User to decide on which individual name to use, or to use the entire set.

5.2.2 Performing Deprecation

When modifying an existing identifier WFN within a CPE dictionary, the Dictionary Maintainer MUST deprecate the existing identifier WFN, and link the deprecated identifier WFN to the set of new identifier WFNs that are replacing it through the deprecated-by relationship[8]. The deprecated-by relationship links a single identifier WFN to one or more disparate identifier WFNs. For example, if x is a single identifier WFN and y is a set of identifier WFNs, then "x deprecated-by y" defines a deprecation relationship between x and y. In this example, x represents a single identifier WFN that the Dictionary Maintainer is deprecating and y represents a set of identifier WFNs that is deprecating x.

Deprecating an identifier WFN signifies to CPE dictionary users that the deprecated name is no longer a valid product identifier. The deprecation of an identifier WFN within a dictionary may occur in three distinct ways according to the type of deprecation required:

1. The Dictionary Maintainer is performing a deprecation of type “Identifier WFN Correction”. In this case, the Dictionary Maintainer will deprecate the identifier WFN and list exactly one unique identifier WFN as replacing it.
2. The Dictionary Maintainer is performing a deprecation of type “Identifier WFN Removal”. In this case, the Dictionary Maintainer will deprecate the identifier WFN and list no new identifier WFNs.
3. The Dictionary Maintainer is performing a deprecation of type “Additional Information Discovery”. In this case, the Dictionary Maintainer will deprecate the identifier WFN and specify the set of identifier WFNs replacing it.

When a Dictionary Maintainer decides to deprecate an identifier WFN, then the Dictionary Maintainer MUST follow the below deprecation process, this process refers to the identifier WFN that the Dictionary Maintainer is deprecating as the *legacy WFN*:

1. If the Dictionary Maintainer is replacing the *legacy WFN* with new identifier WFNs, then the Dictionary Maintainer MUST add the new identifier WFN, or set of identifier WFNs, to the CPE Dictionary.
2. The Dictionary Maintainer MUST add a *deprecation* element to the *legacy WFN* dictionary entry to signify that it is deprecated. If the *legacy WFN* was previously deprecated, then the Dictionary Maintainer MUST add a new *deprecation* element to record the new deprecation[9].
3. The Dictionary Maintainer MUST expand the *deprecation* element from Step 2 to include one or more *deprecated-by* elements to record the identifier WFN(s) replacing the *legacy WFN*. The Dictionary Maintainer MUST specify the type of deprecation within the *type* attribute of the *deprecated-by* element.
4. If the Dictionary Maintainer is performing a deprecation of type “Identifier WFN Correction”, then the Dictionary Maintainer MUST populate the *name* attribute of the *deprecated-by* element with the identifier WFN replacing the *legacy name.*

[8] Section 9 defines the data model that includes the deprecated-by relationship. Specifically the *deprecated-by* relationship is a property of the *deprecation* element that captures all deprecation data for a specific deprecation occurrence.

[9] It is possible that multiple deprecations may occur against a single identifier WFN at different times (e.g. it is discovered that more products are included in the *deprecated-by* set). To support this, the CPE Dictionary Data model requires that each *deprecation* elements within an identifier WFN record the deprecation for one instant in time.

5. If the Dictionary Maintainer is performing a deprecation of type "Additional Information Discovery", then the Dictionary Maintainer MUST populate the *name* attribute of the *deprecated-by* element with a WFN representing the set of new identifier WFNs that are replacing the *legacy name*; this WFN may contains wildcards (e.g. *, ?) to represent a set of identifier WFNs.
6. The Dictionary Maintainer MUST record the change against the *legacy WFN* in the legacy WFN's provenance information.

5.2.3 Use of Deprecated Names

Organizations and tools using identifier WFNs from a CPE dictionary MUST NOT use a deprecated identifier WFN, but MUST instead use an identifier WFN that is linked to the deprecated identifier WFN through the *deprecated-by* relationship[10]. It is important to understand that even though a set of names may deprecate one identifier WFN, an organization does not have to use all of these new names. The organization MAY simply pick a non-deprecated name out of the set, or the organization MAY choose to use the entire set.

Any organization or tool using a deprecated identifier WFN MUST resolve the correct non-deprecated identifier WFNs using the following process, this process references the `dictionary-search` function defined in Figure 10-4:

1. If the deprecated identifier WFN does not reference any identifier WFNs within the *deprecated-by* element, then the organization MUST choose a new non-deprecated identifier WFN from the dictionary. This situation will occur if deprecation in question is of type "Identifier WFN Removal".
2. If the deprecated identifier WFN does provide an identifier WFN within the *deprecated-by* element and the deprecation type is "Identifier WFN Correction" then the organization MUST resolve the dictionary entry containing the identifier WFN listed. The organization MUST resolve this identifier WFN using the `dictionary-search` function, passing the identifier WFN, the dictionary, and a value of *true* as the parameters to the function; these arguments will result in an identifier lookup operation.
3. If the deprecated identifier WFN does provide a WFN within the *deprecated-by* element and the deprecation type is "Additional Information Discovery" then the organization MUST resolve the set of dictionary entries containing the identifier WFNs. The organization MUST resolve this set of identifier WFNs using the `dictionary-search` function, passing the WFN, the dictionary, and a value of *false* as the parameters to the function.
4. It is possible that multiple *deprecation* elements will exist, or that a single *deprecation* element will contain multiple *deprecated-by* elements, normally all of type "Additional Information Discovery". In this case, the organization MUST iterate through the above steps for all *deprecated-by* elements provided. The organization MUST take the union of all resolved sets of identifier WFNs, this union is the correct set of identifier WFNs that have replaced the *legacy WFN*.
5. The organization MAY encounter deprecated identifier WFNs in the set of identifier WFNs resolved in Step 4. In this case, the organization MUST follow the above process replace these deprecated names with the set of names that deprecated it; this process may be recursive.
6. The final set of resolved names represents all names that replace the *legacy WNF*. Organizations MAY either use all of these names, or pick one name out of the set to use in place of the *legacy* WFN.

[10] This requirement does not apply to tools and organization purposefully communicating information relating to deprecated WFNs.

5.3 CPE Dictionary Provenance Data

CPE Dictionary Maintainers MUST capture the required provenance data specified in section 9. The provenance data required includes data useful in understanding the reasoning behind changes made to identifier WFNs stored within a CPE Dictionary. This provenance data also captures the organization responsible for identifier WFN submissions, as well as the authority behind the submissions.

6. CPE Dictionary Searching

This section defines the different ways CPE Dictionary users can perform searching against a CPE Dictionary. There are two scenarios relating to searching against a CPE dictionary. The first scenario involves looking up a single identifier WFN within a dictionary. The second scenario involves using WFN representing a set of products to search against the dictionary to determine what identifier WFNs within the dictionary are members of that set. Both of these scenarios leverage the CPE Matching algorithm[11] when performing the search operations. The following sub-sections explore both scenarios in detail using pseudo code to define the searching algorithms.

6.1 Identifier Lookup

Identifier lookup involves using a single identifier WFN (i.e. the source identifier WFN) to search against a dictionary to see if that identifier exists within the dictionary. In this scenario, a tool would iterate over each entry in a dictionary to determine if any identifier WFN in the dictionary is equal to the source identifier WFN. Equality is determined if all attribute values in the source identifier WFN are equal to the corresponding attribute values of a dictionary identifier WFN. The `CPE_EQUAL` function defined in the CPE Matching Specification [CPE23-M:7.2] presents a formal implementation of this equality test. There are two useful results from this searching operation: either the source identifier WFN matches exactly against one, and only one, dictionary identifier WFN, or there is no match. The following list defines each result type in more detail:

- **Match** – Source identifier WFN matches exactly against one, and only one, identifier WFN within the dictionary. When a match is found the result of the operation will be the dictionary identifier WFN that matched.
- **No Match** – Source identifier WFN does not fully match against any identifier WFN within the dictionary. When no match is found the result of the operation will be a null value.

Section 10.1.3 defines the formal implementation of this identifier lookup operation in abstract pseudo code.

6.2 Dictionary Searching

Dictionary searching involves using a source WFN representing a set of products to search against a dictionary to determine what identifier WFNs within the dictionary are members of that set. This scenario is similar to using a SQL query to search against a relational database to find a specific set of rows; in this analogy, the source WFN represents the SQL query, and the dictionary represents the relational database. This type of searching leverages the CPE Matching algorithm to determine the relationship between the set represented by the source WFN and the identifier WFNs within the dictionary. In dictionary searching, a tool would iterate over each entry in the dictionary to determine if any identifier WFN within the dictionary is a member of the set represented by the source WFN. There are three useful results from this searching operation; the following list defines each result type in more detail:

- **Superset Match** – Set represented by source WFN contains one or more identifier WFNs from the dictionary. In set theory language, the set represented by the source WFN is a superset of a

[11] The CPE Matching Specification [CPE23-M:7.2] formally defines the CPE Matching Algorithms used in this section, and in accompanying pseudo code implementations in section 10.1.3.

portion of the dictionary; containing one or more identifier WFNs within the dictionary. When a match is found the result of the operation will be the set of matching dictionary identifier WFNs.

- **Subset Match** – A Subset condition arises when the set represented by the source WFN is a *possible* subset of one or more identifier WFNs within the dictionary. This situation will only occur if the source WFN it is more specific than one or more names within the dictionary (i.e. represents a subset of a dictionary name). In some situations, this result may raise an error, since the tool/organization performing the search may want to notify the Dictionary Maintainer that a WFN in the wild is more specific than any identifier WFN within the dictionary.
- **No Match** – There is no relationship between the set represented by the source WFN and the identifier WFNs within the dictionary. The source WFN is disjoint with the dictionary. When no match is found, the result of the operations will be a null value.

Section 10.1.3 defines the formal implementation of this dictionary search operation in abstract pseudo code. This pseudo code leverages the `CPE_SUPERSET` and `CPE_SUBSET` functions defined in the CPE Matching specification [CPE23-M:7.2] to implement the search operation.

7. Management Documents

Every CPE dictionary is required to have a set of supporting management documentation associated with it. This set of supporting documentation is required to improve the transparency relating to dictionary acceptance criteria, dictionary content creation decisions and processes associated with the dictionary. When an organization serves as a CPE Dictionary Maintainer it MUST create, or reference, a series of accompanying management documents capturing rules and processes specific to the dictionary maintained by the organization.

It is possible for a single community to establish multiple Extended CPE dictionaries. In these situations the Extended Dictionary Maintainers MAY reference a set of external CPE Dictionary Management documents as the authoritative documents for the specific Extended Dictionary.

The remainder of this section defines each management document required.

7.1 Dictionary Content Management and Decisions Document

Dictionary Maintainers MUST either create or reference a *Dictionary Content Management and Decisions Document*. The purpose of the decisions document is to document the procedures related to CPE identifier creation within a particular dictionary or set of dictionaries. All procedures defined in the decisions document MUST NOT override or conflict with the high level policy defined in the CPE dictionary Specification. The decisions document MUST capture the following information:

1. Rules relating to dictionary specific acceptance criteria for CPE Names.
2. Identification Strategies relating to different product types. Due to the heterogeneous nature of product versioning in the IT industry, multiple disparate strategies for versioning products exist. Where possible, the Dictionary Content Management and Decisions Document should document the different Identification Strategies captured within the dictionary. For example, if a specific product line uses seven digits within its version syntax, then the Decisions Document should document the semantics of each digit within this version syntax[12].
3. Automated identifier WFN creation strategies for specific products. These strategies may include the API calls or functions to call on certain products to populate specific attributes of an identifier WFN.
4. Lists of valid values for specific CPE attributes where appropriate. Valid value lists may be either global, or pertain to specific CPEs. For example, it is possible to have a granular valid value list for the version attribute of a specific vendor and product, without extending the scope of this valid values list to every CPE version attribute within the dictionary.
5. Abbreviation rules for data within specific CPE attributes where appropriate. Abbreviation rules may be either global, or pertain to specific CPEs.
6. Rules relating to any dictionary specific provenance data which the dictionary records.

[12] Appendix B provides an informative overview relating to differences between versioning strategies in the IT industry.

7.2 Dictionary Process Management Document

Dictionary Maintainers MUST either create or reference a *Dictionary Process Management Document*. The purpose of the process management document is to document all processes associated with a particular dictionary or set of dictionaries. All procedures defined in the process management document MUST NOT override or conflict with the high level policy defined in the CPE dictionary Specification. The process management document MUST capture the following information:

1. The scope of the dictionary.
2. The target audience of the dictionary.
3. The submission process for the dictionary that users must follow to submit new CPE identifiers for inclusion within the dictionary. At a minimum, this overview should include the submission format, the process for starting the submission process, and the workflow surrounding the submission process.
4. The content decisions process that the community follows to create the content decision rules that are captured in the Dictionary Content Management and Decisions Document.
5. The CPE Identifier modification process followed by Dictionary Maintainers.
6. The dictionary distribution process and methodology. At a minimum, this should define the binding, in which the dictionary is distributed.

8. Official and Extended Dictionaries

A distinction exists between the Official CPE Dictionary and Extended CPE dictionaries. This section defines each concept and clarifies the distinction between the two.

8.1 Official CPE Dictionary

The National Institute of Standards and Technology hosts the Official CPE Dictionary[13], which is the authoritative repository of identifier CPE names. The goal of the CPE stack of specifications is to provide entities within the IT industry a standardized way to describe and identify IT products. The Official CPE Dictionary provides the mechanism to support this interoperability for product identifiers. The authoritative nature of the Official CPE Dictionary allows organizations to search for, and find identifier WFNs in one centralized place without worrying about dealing with conflicts between federated dictionaries.

The Official CPE Dictionary MUST be fully compliant with the requirements of this specification. Specifically the Official CPE Dictionary MUST meet the following requirements:

- The Official CPE Dictionary MUST be conformant with all organizational conformance rules in section 3.2
- The Official CPE Dictionary MUST house its own set of management documents as specified in section 7
- While permitted, the Official CPE Dictionary SHOULD NOT restrict the acceptance criteria defined in section 5.1 because doing so would require all Extended CPE dictionaries to implement the same restrictions.

8.2 Extended CPE Dictionaries

Organizations MAY stand up Extended CPE dictionaries to store identifier WFNs not present in the Official CPE Dictionary. An Extended CPE dictionary is a dictionary that an organization may stand up to house identifier WFNs not found in the Official CPE Dictionary. There are multiple reasons for possible inconsistency between the official dictionary and extended dictionary identifier set. For example, an organization may have to create identifier WFNs for proprietary products that are only useful within that organization. Another organization may track new products which do not yet have official identifier WFNs, this organization may want to track these identifier WFNs internally until the identifier WFN quality has reached a point where the organization submits them to the Official CPE Dictionary.

Organizations creating Extended CPE dictionaries MUST adhere to the following requirements:

- Extended CPE dictionaries MUST be conformant with the organizational conformance rules in section 3.2
- Extended CPE dictionaries SHOULD house its own set of management documents as specified in section 7. If organic documents are not required, the Dictionary Maintainer SHOULD reference the Official CPE Dictionary Management Documents as applicable to the extended dictionary.

[13] The Official CPE Dictionary is available at http://nvd.nist.gov/cpe.cfm.

- Extended CPE dictionaries MUST adhere to any acceptance criteria restrictions implemented in the Official CPE Dictionary.
- Extended CPE dictionaries MAY further restrict the Official CPE Dictionary acceptance criteria, but it MUST not conflict with the official dictionary criteria.
- Extended CPE dictionaries SHOULD NOT contain identifier WFNs that conflict with the Official CPE Dictionary. This means that if both the Official CPE Dictionary and an Extended CPE dictionary contain an identifier WFN for the same product, then that identifier WFN SHOULD be the same. If Extended Dictionary Maintainers do contain conflicting names, they SHOULD try to resolve the confliction with the Official Dictionary.
- Extended CPE dictionaries MUST NOT include non-unique WNFs with respect to matching. Section 5.1.4 defines this rule in respect to a single dictionary, Extended Dictionaries MUST NOT contain non-unique identifier WFNs with respect to itself or the Official CPE Dictionary.
- Organizations MAY use Extended CPE dictionaries to store identifier WFNs for proprietary products.
- Organizations MAY use Extended CPE dictionaries to store identifier WFNs not yet found in the Official CPE Dictionary, but these organizations SHOULD try to submit these new identifier WFNs to the Official CPE Dictionary as official identifier WFNs.

9. Data Model Overview

This section defines the data model that all CPE Dictionaries MUST implement. The data model does not prescribe a specific binding or implementation. It merely describes the data that is required to support the technical use cases. This section uses the term "*element*" to identify the classes within the data model, and the term "*property*" to identify any properties of a class. Values of a property may include a simple literal value, or another element representing some complex relation to the top-level element; to clarify this distinction the type column lists the type of the property. The syntax for literal types is "*literal - type*", where *type* is the specific literal type. The syntax for element types is "*element - type*", where *element* is the specific element type.

This data model makes special accommodations to ensure that bindings of the data model may remain backwards compatible with the CPE Dictionary 2.2 XML Schema. Any XML Schema based bindings generated from this data model may produce instance data that validates against the CPE Dictionary 2.2 XML Schema.

A CPE dictionary is only a collection of identifier WFNs and metadata associated with these identifiers. To support this, the CPE Dictionary data model revolves around one core element called "*cpe-item*" that holds all the information relating to a single identifier WFN. The *cpe-item* element has evolved out of the CPE Dictionary 2.2 XML Schema to support backwards compatibility. The *cpe-item* element contains a *"cpe23-item*" element that captures all CPE 2.3 specific data including provenance data and an upgraded deprecation system.

9.1 Mandatory Elements

This section defines the mandatory elements for each identifier WFN within a CPE Dictionary.

Element Name: cpe-list				
Definition	This element contains all cpe-items held within a dictionary			
cpe-item properties	**Name**	**Type**	**Count**	**Definition**
	cpe-item	element - cpe-item	1-n	This element represents a single identifier WFN within a CPE Dictionary. All metadata relating to the specific identifier WFN is contained within this element.
Example Binding (Text)	`cpe-item: see cpe-item element`			

Example Binding (XML)	`<cpe23:cpe-list>`
	`  <cpe23:cpe-item name="cpe:/a:adobe:acrobat:3”>`
	`  ....`
	`  </cpe23:cpe-item>`
	`</cpe23:cpe-item>`			

Element Name: cpe-item

Definition	This element represents a single identifier WFN within a CPE Dictionary. All metadata relating to the specific identifier WFN is contained within this element.			
cpe-item properties	**Name**	**Type**	**Count**	**Definition**
	name	literal - CPE 2.2 URI	1	The CPE 2.2 version of the WFN in the CPE 2.2 URI binding.
	title	literal - string	1-n	Human readable title of the WFN.
	cpe23-item	element - cpe23-item	1	Element that captures all CPE 2.3 specific data including the CPE 2.3 formatted string binding of the WFN.

Example Binding (Text)	`cpe-item: cpe:/a:adobe:acrobat:3`
	`title: Adobe Acrobat 3`
	`cpe23-item: *see cpe23-item element*`			

Example Binding (XML)	`<cpe23:cpe-item name="cpe:/a:adobe:acrobat:3”>`
	`<cpe23:title xml:lang="en-US">Adobe Acrobat 3</title>`
	`<cpe23:cpe23-item name="cpe23:a:adobe:acrobat:3:*:*:*:*:*:*:*"`
	`.....`
	`</cpe23:cpe23-item>`
	`</cpe23:cpe-item>`			

874
875

Element Name: cpe23-item				
Definition	This element captures all CPE 2.3 specific data including the CPE 2.3 formatted string binding of the WFN, provenance data, and deprecation data.			
cpe-item properties	**Name**	**Type**	**Count**	**Definition**
	name	literal - CPE 2.3 formatted string	1	The CPE 2.3 version of the identifier WFN in the CPE 2.3 formatted string binding.
	provenance-record	element - provenance-record	1	Element holding all provenance information for the given identifier WFN.
	deprecation	element - deprecation	0-n	Element holding one or more deprecation entries for the given identifier WFN. It is possible for a single identifier WFN to have multiple deprecations that occur at different time periods.

Example Binding (Text)	`cpe23-item: cpe23:a:adobe:acrobat:3:*:*:*:*:*:*:*`
	`provenance-record: *see provenance-record element*`
	`deprecation: *see deprecation element*`			

Example Binding (XML)	`<cpe23:cpe23-item name="cpe23:a:adobe:acrobat:3:*:*:*:*:*:*:*">`
	`<cpe23:provenance-record>`
	`....`
	`</cpe23:provenance-record>`
	`<cpe23:deprecation date="2006-05-04T18:13:51.0Z">`
	`....`
	`</cpe23:deprecation>`
	`</cpe23:cpe23-item>`			

876
877

Element Name: provenance-record				
Definition	Element holding all provenance information for the given identifier WFN.			
cpe-item properties	**Name**	**Type**	**Count**	**Definition**
	submitter	element - organization	1	The organization responsible for submitting the identifier WFN.
	authority	element - organization	1-n	The authority responsible for endorsing the identifier WFN. Multiple authorities may endorse the same identifier WFN.
	change-description	element – change-description	1-n	A description of any changes made to the identifier WFN.

Example Binding (Text)	`submitter: see organization element`
	`authority: see organization element`
	`change-history: see change-history element`			

Example Binding (XML)	`<cpe23:provenance-record>`
	`  <cpe23:submitter system-id="http://nvd.nist.gov/" name="NVD" date="2006-05-04T18:13:51.0Z"/>`
	`  <cpe23:authority system-id="http://nvd.nist.gov/" name="NVD" date="2006-05-04T18:13:51.0Z"/>`
	`  <cpe23:change-description change-type="ORIGINAL_RECORD" date="2006-05-04T18:13:51.0Z">`
	`  ....`
	`  </cpe23:change-description>`
	`  <cpe23:change-description change-type="DEPRECATION" date="2007-05-04T18:13:51.0Z">`
	`  ....`
	`  </cpe23:change-description>`
	`</cpe23:provenance-record>`			

878
879

Element Name: organization				
Definition	Element holding information about a specific organization.			
cpe-item properties	**Name**	**Type**	**Count**	**Definition**
	system-id	literal - URI	1	Unique URI representing the organization.
	name	literal - String	1	Human readable name of the organization.
	date	literal - Date/Time	1	The date the organization performed an action relative to an identifier WFN. For example, the date the organization submitted, or endorsed a particular identifier WFN.

Example Binding (Text)	`name: NVD`
	`system-id: http://nvd.nist.gov/`
	`change-history: see change-history element`
	`date: 2006-05-04`			

Example Binding	`<cpe23:submitter system-id="http://nvd.nist.gov/" name="NVD" date="2006-05-04T18:13:51.0Z"/>`			

(XML)	

880
881

Element Name: change-description				
Definition	A description of any changes made to the identifier WFN, or associated metadata.			
cpe-item properties	**Name**	**Type**	**Count**	**Definition**
	change-type	literal - String	1	The type of change that occurred. The value for this property MUST be one of the following values: "*ORIGINAL_RECORD*", "*DEPRECATION*", "*AUTHORITY_CHANGE*" "*DEPRECATION_MODIFICATION*". The meaning of these values is defined below: • *ORIGINAL_RECORD* – This change type should be used when the WFN is first added to the dictionary. • *AUTHORITY_CHANGE* – This change type should be used when the authority behind the identifier WFN is modified. • *DEPRECATION* – This change type should be used when the WFN is first deprecated. • *DEPRECATION_MODIFICATION* – This change type should be used when additional deprecation entries are recorded for a deprecated WFN.
	date	literal – Date/Time	1	Date when the change occurred.
	comments	literal - String	0	Comments explaining the rationale for the change.
	evidence-reference	element – evidence-reference	0	Supporting evidence for the change including a link to external information relating to the change.

Example Binding (Text)	`change-type: DEPRECATION`
	`date: 2007-05-04`
	`evidence-reference: see` *`evidence-reference`* `element`
	`comments: This name was deprecated`			

Example Binding (XML)	`<cpe23:change-description change-type="DEPRECATION" date="2007-05-04T18:13:51.0Z">`
	`  <cpe23:evidence-reference evidence="CURATOR_UPDATE">`
	`  http:/adobe.com/versionHistory`
	`  </cpe23:evidence-reference>`
	`  <cpe23:comments>This name was deprecated</cpe23:comments>`
	`</cpe23:change-description>`			

882
883
884

885 **9.2 Optional Elements**

Element Name: deprecation				
Definition	An element containing information for a specific deprecation of an identifier WFN. A single deprecation element may contain a list of WFNs that the enclosing identifier WFN was deprecated by. One deprecation element represents a deprecation that occurred at a specific instant in time, it is possible that additional deprecations will occur at a later instant in time. If a Dictionary Maintainer must submit deprecation entries after the initial deprecation, then another deprecation element should be added to the identifier WFN.			
cpe-item properties	**Name**	**Type**	**Count**	**Definition**
	date	literal – Date / Time	1	The date the deprecation entry was entered.
	deprecated-by	element - deprecated-by	1-n	The element containing the list of WFNs that deprecated the enclosing identifier WFN. The WFN contained within this property does not have to represent an identifier WFN. This means that the WFN may contain wildcards (e.g. ANY, *, ?) and may represent a set of products. This provides a more robust mechanism to support One to Many deprecation logic.
Example Binding (Text)	`deprecation date: 2006-05-04` `deprecated-by: see` ***`deprecated-by`*** `element`			

Example Binding (XML)	`<cpe23:deprecation date="2006-05-04T18:13:51.0Z">`
	`  <cpe23:deprecated-by name="cpe23:a:adobe:acrobat:3.0:*:*:*:*:*:*:*"`
	`  type="NAME_CORRECTION"/>`
	`</cpe23:deprecation>`			

886

Element Name: deprecated-by				
Definition	The element containing the list of WFNs that deprecated the enclosing identifier WFN. The WFN contained within this element does not have to represent an identifier WFN. This means that the WFN may contain wildcards (e.g. ANY, *, ?) and may represent a set of products. This provides a more robust mechanism to support One to Many deprecation logic.			
cpe-item properties	**Name**	**Type**	**Count**	**Definition**
	name	literal – CPE WFN	1	The WFN that is deprecating the containing cpe-item.
	type	literal - Boolean	1	The type of deprecation associated with the deprecated-by element. The value for this property MUST be one of the following values: "*NAME_CORRECTION*", "*NAME_REMOVAL*", "*ADDITIONAL_INFORMATION*" The meaning of these values is defined below: • *NAME_CORRECTION* – Specifies the deprecation is of type "Identifier WFN Correction" • *NAME_REMOVAL* – Specifies the

				deprecation is of type "Identifier WFN Removal" • *ADDITIONAL_INFORMATION* – Specifies the deprecation is of type "Additional Information Discovery" These types are defined in more detail in section 5.2.1.
Example Binding (Text)	`name: cpe23:a:adobe:acrobat:3.0:*:*:*:*:*:*:*` `type: NAME_CORRECTION`			
Example Binding (XML)	`<cpe23:deprecated-by name="cpe23:a:adobe:acrobat:3.0:*:*:*:*:*:*:*" type="NAME_CORRECTION"/>`			

887
888

Element Name: evidence-reference				
Definition	Supporting evidence for any change to a WFN, or associated metadata, including a link to external information relating to the change.			
cpe-item properties	**Name**	**Type**	**Count**	**Definition**
	url	literal – url	1	The URL referencing a specific piece of evidence.
	evidence	literal – Sting	1	The type of evidence the given URL provides. The value for this property MUST be one of the following values: "*CURATOR_UPDATE*", "*VENDOR_FIX*", "*THIRD_PARTY_FIX*". The meaning of these values is defined below: • *CURATOR_UPDATE* – The curator of the dictionary discovered information that led to a change. • *VENDOR_FIX* – The vendor of the product identified in the enclosing WFN released, or submitted, information that led to a change. • *THIRD_PARTY_FIX* – A third party released, or submitted, information that led to a change.
Example Binding (Text)	`evidence: CURATOR_UPDATE` `url: http:/adobe.com/versionHistory`			

Example Binding (XML)	`<cpe23:evidence-reference evidence="CURATOR_UPDATE">`
	`  http:/adobe.com/versionHistory`
	`</cpe23:evidence-reference>`			

889
890
891

9.3 Extension Points

Organizations may need to capture data not defined in the CPE Dictionary data model; any organization serving as a CPE Dictionary Maintainer MAY extend the *cpe23-item*, *cpe-list* and the *provenance-record* element to capture additional, organization specific data. If organizations must extend these elements, this extension MUST only occur by adding additional properties to these elements to capture different types of data, or by restricting the values for specific properties. Organizations MUST NOT define rules conflicting with the properties already defined for these elements.

10. Implementation and Binding

10.1 CPE Dictionary Pseudo Code

The following sub-sections contain algorithms that implement concepts described in the rest of the specification. This specification uses an abstract pseudo-code programming language to specify intended computational behavior. Pseudo-code is intended to be straightforwardly readable and translatable into real programming language terms. In reading pseudo-code the following notes should be kept in mind:

- All pseudo-code functions are *pass by reference*, meaning that any changes applied to the supplied arguments within the scope of the function do not affect the values of the variables in the caller's scope.
- In a few cases, the pseudo-code functions reference (more or less) standard library functions, particularly to support string handling. Whenever possible, we reference semantically equivalent functions from the GNU C library, (cf. http://www.gnu.org/software/libc/manual/html_node/index.html#toc_String-and-Array-Utilities).

10.1.1 Operations on a CPE Dictionary

This section defines a set of functions for performing common activities against a CPE Dictionary. These functions are relatively simple and do not require a pseudo code implementation, but the remaining sub-sections will present pseudo code that may utilize these common functions.

10.1.1.1 Function get_cpe_items(d)

The `get_cpe_items(d)` function takes a single CPE Dictionary, `d`, and returns back all `cpe-items` associated with it. A single `cpe-item` within the dictionary represents the identifier WFN, and all associated metadata[14].

10.1.1.2 Function get_cpe_item_WFN(item)

The `get_cpe_item_WFN(item)` function takes a single `cpe-item` element and returns back the identifier WFN that it represents.

10.1.1.3 Function get(w,a)

The `get(w,a)` accessor function takes two arguments, a WFN *w* and an attribute *a*, and returns the value of *a*. This function is officially defined in the CPE Naming Specification [CPE23-N:5.6.2].

10.1.1.4 Function is_deprecated(item)

The `is_deprecated(item)` function takes in a single `cpe-item` and returns back `true` if the `cpe-item` is deprecated, `false` if the `cpe-item` is not deprecated.

[14] Section 9.1defines the `cpe-item` element in detail.

10.1.1.5 Function getItem(list, index)

The `getItem(list, index)` function is a helper function for retrieving an item in a list. The function will return the list item at the position specified by `index`. This function assumes a 0-based index and will return `null` if no items exist at the provided index.

10.1.2 Acceptance Criteria Pseudo Code

This section defines the algorithm required to implement the acceptance criteria defined in section 5.1. The core algorithm is implemented in the below pseudo code function named `accept-name`, that processes a given WFN, `w`, and compares it against a specific dictionary to determine if the dictionary should accept the new name. The following list provides a brief summary of the algorithm implemented in the `accept-name` function:

1. Use the helper function `contains-restricted-characters` to determine if the WFN `w` contains any of the restricted characters defined in section 5.1.2. If `w` contains any of the restricted characters, `accept-name` will return false.
2. Use the helper function `contains-required-attributes` to determine if the WFN `w` contains known data for all of the required attributes specified in section 5.1.3. If `w` does not contain all of the required attributes, `accept-name` will return false.
3. Use the helper function `matches-more-complete-in-dictionary` to determine if the WFN `w` is unique within the dictionary `d`, as specified in section 5.1.4. If `w` does not adhere to this rule, or in other words if w matches against a more complete name in the given dictionary `d`, then `accept-name` will return false. This function uses the `dictionary-search` function defined in section 10.1.3 for matching against the dictionary.

```
1  function accept-name(w, d)
2    ;; Top-level function to determine if the WFN CPE w should be
3    ;; accepted into dictionary d based on high-level acceptance
4    ;; criteria.  Assumes WFN meets acceptance criteria defined in CPE
5    ;; Naming Spec.
6    if contains-restricted-characters(w)
7      then return false.
8    endif.
9    if !contains-required-attributes(w)
10     then return false.
11   endif.
12   if matches-more-complete-in-dictionary(w, d)
13     then return false.
14   endif.
15   return true.
16 end.
```

Figure 10-1: accept-name function

```
1  function contains-restricted-characters(w)
2    ;; Helper-function to determine if WFN CPE w contains characters
3    ;; not permitted in dictionary.
```

```
4    foreach a in w do  ;; loop through every attribute in WFN
5      s := get(w,a) ;; get string value of attribute
6      n := 0.
7      loop
8        if n >= strlen(s)
9          then break. ;; break to outer loop
10       endif.
11       c := substr(s,n,1).  ;; get the n'th character of s.
12       if ((c = "*" or c = "?") and (substr(s,n-1,1) != "\"))
13         then
14           if (c = "*" and n = 0 and (n + 1 < strlen(s)))
15             then continue.    ;; single '*' represents ANY
16             else return true. ;; embedded '*' or '?' not permitted
17           endif.
18         else
19           ;; character is legal, move on
20           n := n + 1.
21           continue.
22       endif.
23     endloop.
24   endfor.
25   return false.
26 end.
```

Figure 10-2: contains-restricted-characters function

```
1  function contains-required-attributes(w)
2    ;; Helper-function to determine if required attributes contain
3    ;; known data.  WFN syntax defined in CPE Naming Spec ensures all
4    ;; attributes contain at least some data.
5    foreach a in {part, vendor, product, version} do
6      s := get(w,a) ;; get string value of attribute
7      ;; only loop through required attributes of w
8      if s = "ANY"
9        then return false.
10     endif.
11     if (a != version and s = "NA")
12       then return false. ;; NA only permitted in version
13     endif.
14   endfor.
15   return true.
16 end.
```

Figure 10-3: contains-required-attributes function

```
1  function matches-more-complete-in-dictionary(w, d)
2    ;; Helper-function to determine if identifier WFN w matches a
3    ;; more complete name in the dictionary d (i.e. a superset match).
4    matches := dictionary-search(w, d, false)
5    if (size(matches) > 0)
6      then
```

```
7        if (getItem(matches, 0) = "SUPERSET-MATCH")
8          then return false.  ;;at least one superset match was found
9        endif.
10   endif.
11   return true.  ;; no match, or subset match are both okay.
12 end.
```

10.1.3 Dictionary Searching Pseudo Code

This section Figure 10-4 contains pseudo code that implements the identifier lookup and dictionary search operations defined in section 6. The *dictionary-search* function implements the two searching scenarios defined in sections 6.1 and 6.2[15]. This function is focused on returning all non-deprecated names within a dictionary, but will not filter out deprecated names from the result set, allowing the caller to filter the names based on use case. While this function will not filter out deprecated names, it does not guarantee that it will return all matching deprecated names in the result set; it only guarantees the return of all matching non-deprecated names. When not looking for exact matches, the function will look for all superset matches within the given dictionary. The function will only look for subset matches if the given dictionary contains no superset matches. This ordering derives from the CPE Dictionary acceptance criteria that will not allow a dictionary to contain non-deprecated subset matches if a superset match is present[16].

Table 10-1 provides a detailed overview of the `dictionary-search` function. This pseudo code leverages the accessor functions defined in section 10.1.1. The code also leverages the `CPE_EQUAL, CPE_SUBSET, and CPE_SUPERSET` functions defined in The CPE Matching Specification [CPE23-M:7.2]

Table 10-1: Description of dictionary-search function

Line Number(s)	Description
1	The function accepts three arguments, the source WFN, the dictionary to search against, and a boolean flag 'exact'. When 'exact' is true the function will perform and identifier lookup based on the source WFN. When 'exact' is false the function will perform dictionary searching for all names contained the source WFN set.
11	Creates a list to store all discovered matches as well as the match type; if a match is found, then the first position of this list will contain the match type and the second position will contain the matches (either single match, or set of matches).
12 -13	If 'exact' is true, the function will enter into logic to determine if an exact match is present within d. The function will then call the findExactMatch function (defined in Figure 10-7) to determine if an exact match exists in the dictionary.
14-20	If an exact match is found, the function will populate the 'response' list with the response type of 'EXACT-MATCH' and then append the exact match to the next position in the list. The function will then return

[15] This function does not account for deprecation chains. For example if an identifier WFN is found that is deprecated, a tool may want to recursively search all the identifier WFNs in the deprecation chain (i.e. the set of all identifier WFNs referenced by the deprecated-by property.

[16] The `dictionary-search` function was designed for readability; more efficient methods for implementing this logic exist.

Line Number(s)	Description
	the response list, or null if no match was found.
22	The function will call the findSupersetMatches function (defined in Figure 10-5) to build the set of all superset matches found in the d.
23-28	If any superset matches are found, the function will populate the 'response' list with the response type of 'SUPERSET-MATCH' and then append the set of matches to the next position in the list. The function will then return the response list.
29	The function will call the findSubsetMatches function (defined in Figure 10-6) to build the set of all subset matches found in the d. NOTE: No non-deprecated subset matches should be present if a superset match was found.
30-35	If any subset matches are found, the function will populate the 'response' list with the response type of 'SUBSET-MATCH' and then append the set of matches to the next position in the list. The function will then return the response list.
36	If no matches were found the function will return null.

```
1  function dictionary-search(source, d, exact)
2    ;; For a given source WFN, source, the function will determine
3    ;; how it relates to the given dictionary, d.  If the source WFN
4    ;; is a superset match to one or more identifier WFNs in d, then
5    ;; the function will return that set of names.  If the source name
6    ;; is not a superset of any dictionary names, but is a subset of a
7    ;; dictionary name(s), then the function will return the set of
8    ;; dictionary names of which it is a subset. If an exact match is
9    ;; found and exact is true the function will return the exact
10   ;; match.  If no match exists the function will return null.
11   response := new List().
12   if (exact = true)
13     match := findExactMatch(source, d).
14     if (match != null)
15       then
16         response := append(response, "EXACT-MATCH").
17         response := append(response, match)
18         return response.
19       else return null.
20     endif.
21   endif.
22   supersetMatches := findSupersetMatches(source, d).
23   if (size(supersetMatches) > 0)
24     then  ;;superset matches found
25       response := append(response, "SUPERSET-MATCH").
26       response := append(response, supersetMatches)
27       return response.
28   endif.
29   subsetMatches := findSubsetMatches(source, d).
30   if (size(subsetMatches) > 0)
31     then  ;; subset matches found
32       response := append(response, "SUBSET-MATCH").
33       response := append(response, subsetMatches).
```

```
34       return response.
35     endif.
36     return null.
37 end
```

Figure 10-4: dictionary-search function

Table 10-2: Description of findSupersetMatch function

Line Number(s)	Description
4	Creates a set to store all discovered superset matches.
5	Starts looping over every cpe-item in given dictionary, d.
7	Retrieves the WFN represented by the current item from the dictionary.
8-11	Passes the source and dictionary WFN to the CPE_SUPERSET function (defined in CPE Name Matching Specification [CPE23-M:7.2] to determine how the two names relate. If the source is a 'superset' of the dictionary name then the function will append the item to the set of matches.
13	The function will return the set of matches

```
1  function findSupersetMatches(source, d)
2  ;; For a given source WFN, source, the function will find all
3  ;; superset matches contained within the given CPE dictionary, d.
4    matches := new Set().
5    foreach item in get_cpe_items(d)
6      do
7        dictionaryName := get_cpe_item_WFN(item).;;WFN from cpe-item
8        if (CPE_SUPERSET(source, dictionaryName) = TRUE
9          then
10           matches := append(matches, item).
11       endif.
12   endfor.
13   return matches.
14 end
```

Figure 10-5: findSupersetMatches function

Table 10-3: Description of findSubsetMatch function

Line Number(s)	Description
4	Creates a set to store all discovered subset matches.
5	Starts looping over every cpe-item in given dictionary, d.
7	Retrieves the WFN represented by the current item from the dictionary.
8-11	Passes the source and dictionary WFN to the CPE_SUBSET function (defined in CPE Name Matching Specification [CPE23-M:7.2] to determine how the two names relate. If the source is a 'subset' of the dictionary name then the function will append the item to the set of matches.
13	The function will return the set of matches

```
1  function findSubsetMatches(source, d)
2  ;; For a given source WFN, source, the function will find all
```

```
3  ;; subset matches contained within the given CPE dictionary, d.
4    matches := new Set().
5    foreach item in get_cpe_items(d)
6      do
7        dictionaryName := get_cpe_item_WFN(item). ;;WFN from cpe-item
8        if (CPE_SUBSET(source, dictionaryName) = TRUE
9          then
10           matches := append(matches, item).
11       endif.
12   endfor.
13   return matches.
14 end
```

Figure 10-6: findSubsetMatches function

Table 10-4: Description of findExactMatch function

Line Number(s)	Description
4	Starts looping over every cpe-item in given dictionary, d.
6	Retrieves the WFN represented by the current item from the dictionary.
7-10	Passes the source and dictionary WFN to the CPE_EQUAL function (defined in CPE Name Matching Specification [CPE23-M:7.2] to determine how the two names relate. If the two names are equal then the function will return the item as the exact match.
12	If not exact matches were found the function will return null.

```
1  function findExactMatch(source, d)
2  ;; For a given source WFN, source, the function will find the
3  ;; exact matche contained within the given CPE dictionary, d.
4    foreach item in get_cpe_items(d)
5      do
6        dictionaryName := get_cpe_item_WFN (item). ;;WFN from cpe-item
7        if (CPE_EQUAL(source, dictionaryName) = TRUE
8          then
9            return item.
10       endif.
11   endfor.
12   return null.
13 end
```

Figure 10-7: findExactMatch function

10.2 CPE Dictionary Binding

Section 9 defines the core data model that CPE Dictionary Creators must use when creating a CPE dictionary. Section 9 does not define a specific binding technology for representing either the data model or instance data generated from the data mode. This separation of data model and binding will allow disparate bindings to evolve when necessary.

Any CPE dictionary binding must adhere to the following requirements:

- CPE dictionary binding MUST implement all data model elements and properties defined in Section 9.

- CPE dictionary binding MAY extend the data model defined in Section 9, but extensions MUST only occur at the valid extension points defined in section 9.3.

Appendix C— lists all valid CPE Dictionary data model bindings.

Appendix A—Use Cases

Using Authoritative Product Identifiers

This specification defines the concept of an Official CPE Dictionary as the repository of authoritative identifier WFNs. This official dictionary provides the mechanism for interoperability across the CPE community since it stores community vetted, authoritative identifier WFNs. When tools or organizations use the identifier WFNs from the Official CPE Dictionary it is expected that other CPE conformant organizations will process these identifiers and interpret them in the same way as the originating organization.

Extending the Official CPE Dictionary

This specification defines the concept of an Official CPE Dictionary as the repository of authoritative identifier WFNs. This official dictionary provides the mechanism for interoperability across the CPE community since it stores community vetted, authoritative identifier WFNs. When tools or organizations use the identifier WFNs from the Official CPE Dictionary it is expected that other CPE conformant organizations will process these identifiers and interpret them in the same way as the originating organization.

The concept of an authoritative, community-vetted dictionary presupposes the need for a detailed process for accepting and maintaining authoritative identifier WFNs. Some organizations may require a CPE identifier management process that is more robust than the process associated with the Official CPE Dictionary. This specification defines the concept of *Extended CPE Dictionaries* to support the communities need to allow organizations to control their own CPE management process.

Organizations wishing to expand the identifier management process associated with the Official CPE Dictionary may choose to stand up internal Extended CPE Dictionaries. Organizations may choose to do this for a variety of reasons including:

- Organizations may need to track, and identify proprietary products that do not belong in the Official CPE Dictionary. These products include any product the organization develops internally to satisfy some organization objective. These products are not found outside the specific organization and therefore a community identifier for these products will serve no value to the community. Organizations with this requirement may stand up an Extended CPE Dictionary to identify/track these proprietary products in the same way as any product listed in the Official CPE Dictionary.

- Organizations may discover new products at a faster rate than the Official CPE Dictionary can process them. Any organization responsible for running IT security operations may discover new IT products at a very fast rate; new here means products that do not yet have official identifier WFNs within the Official CPE Dictionary. In these situations, an organization may stand up an Extended CPE Dictionary, or set of Extended CPE Dictionaries to develop, and track, identifier WFNs for newly discovered products. These organizations would submit these new identifier WFNs when they are satisfied with the level of information captured within the CPE. This process will allow organizations to create new identifier WFNs when found and use them in internal processes even before they are included in the Official CPE Dictionary. When these identifiers are eventually included to the Official CPE Dictionary, the discovering organization will not have to alter its process since it was already using the correct identifier.

Appendix B—Identification Strategies

An *identification strategy* is the way in which a particular product is identified. More specifically this includes the way the product is versioned, the syntax of a version string for a product, and the semantics implied by that version string. Many disparate identifications strategies exist in the IT industry and as a result of this heterogeneousness it is very difficult to create a common syntax to capture the identifying characteristics of a product.

The following IT product examples, from a variety of different vendors illustrates the disparate types of product identification strategies:

- Cisco IOS 12.3(1.2)T
- Cosminexus Application Server 05-01-/A
- firmware designations like 3Com 4500 series switches like V3.01.00s168p01

The CPE Dictionary Specification provides the concept of a Dictionary Content Management and Decisions Document to allow CPE dictionaries to record the different ways in which products are identified. This document provides Dictionary Maintainers a mechanism to record the semantics of the version on a per product basis if necessary. For example, for Cisco IOS the CPE Official Dictionary Content Management Document may record what each digit in the version actually means (e.g. major version, minor version, release, interim build number, train identifier). This way, when users of the dictionary process Cisco IOS WFNs they will have a mechanism that allows them to understand what each digit of the version attribute for the Cisco IOS product actually means.

Appendix C—Valid CPE Dictionary Data Model Bindings

XML Schema Binding

Figure 10-8 captures the CPE 2.2 Dictionary Schema defined in the CPE 2.2 Specification. All 2.2 CPE dictionary content validates against this 2.2 schema. Figure 10-9 defines the CPE 2.3 extension of the 2.2 schema. This extension implements the majority of the CPE Dictionary 2.3 data model defined in section 9. All instance documents created using the 2.3 extension schema will also validate against the CPE 2.2 dictionary schema if the `cpe23-item` is used as the 'any' element in the CPE 2.2 Dictionary Schema `ItemType`. Figure 10-10 captures a sample CPE 2.3 Dictionary instance document that validates against both the CPE 2.2 Dictionary schema, and the 2.3 Dictionary extension schema.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsd:schema targetNamespace="http://cpe.mitre.org/dictionary/2.0"
3        xmlns:cpe_dict="http://cpe.mitre.org/dictionary/2.0"
4        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5        xmlns:xml="http://www.w3.org/XML/1998/namespace"
6        elementFormDefault="qualified"
7        attributeFormDefault="unqualified">
8
9     <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
10         schemaLocation="http://www.w3.org/2001/xml.xsd"/>
11
12    <xsd:annotation>
13         <xsd:documentation xml:lang="en">This is an XML Schema
14              for the CPE Dictionary. It is used to transfer a
15              collection of official CPE Names along with any
16              necessary supporting information (title, references,
17              automated check, etc.). For more information, consult
18              the CPE Specification document.
19         </xsd:documentation>
20         <xsd:appinfo>
21              <schema>CPE Dictionary</schema>
22               <author>Neal Ziring, Andrew Buttner</author>
23               <version>2.2</version>
24               <date>03/11/2009 09:00:00 AM</date>
25         </xsd:appinfo>
26    </xsd:annotation>
27 <!-- ========================================================== -->
28 <!-- ========================================================== -->
29 <!-- ========================================================== -->
30    <xsd:element name="cpe-list" type="cpe_dict:ListType">
31         <xsd:annotation>
32              <xsd:documentation xml:lang="en">
33                   The cpe-list element acts as a top-level
34                   container for CPE Name items. Each individual
35                   item must be unique. Please refer to the
36                   description of ListType for additional
37                   information about the sturcture of this
38                   element.
```

```
            </xsd:documentation>
        </xsd:annotation>
        <xsd:key name="itemURIKey">
            <xsd:selector xpath="./cpe_dict:cpe-item"/>
            <xsd:field xpath="@name"/>
        </xsd:key>
    </xsd:element>
    <xsd:element name="cpe-item" type="cpe_dict:ItemType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                The cpe-item element denotes a single CPE Name.
                Please refer to the description of ItemType for
                additional information about the structure of
                this element.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:unique name="titleLangKey">
            <xsd:selector xpath="./cpe_dict:title"/>
            <xsd:field xpath="@xml:lang"/>
        </xsd:unique>
        <xsd:unique name="notesLangKey">
            <xsd:selector xpath="./cpe_dict:notes"/>
            <xsd:field xpath="@xml:lang"/>
        </xsd:unique>
        <xsd:unique name="checkSystemKey">
            <xsd:selector xpath="./cpe_dict:check"/>
            <xsd:field xpath="@system"/>
        </xsd:unique>
    </xsd:element>
<!-- ========================================================== -->
<!-- =================== SUPPORTING TYPES ===================== -->
<!-- ========================================================== -->
    <xsd:complexType name="GeneratorType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                The GeneratorType complex type defines an element
                that is used to hold information about when a
                particular document was compiled, what version of
                the schema was used, what tool compiled the
              document, and what version of that tools was
            used. Additional generator information is also
            allowed although it is not part of the official
            schema.  Individual organizations can place
            generator information that they feel are
            important and these will be skipped during the
            validation. All that this schema really cares
            about is that the stated generator information is
              there.
        </xsd:documentation>
    </xsd:annotation>
       <xsd:sequence>
        <xsd:element name="product_name" type="xsd:string"
```

```
91                minOccurs="0" maxOccurs="1">
92                     <xsd:annotation>
93                          <xsd:documentation xml:lang="en">
94                          The optional product_name element
95                          specifies the name of the application
96                          used to generate the file.
97                          </xsd:documentation>
98                     </xsd:annotation>
99           </xsd:element>
100               <xsd:element name="product_version" type="xsd:string"
101                    minOccurs="0" maxOccurs="1">
102                    <xsd:annotation>
103                         <xsd:documentation xml:lang="en">
104                              The optional product_version element
105                              specifies the version of the
106                              application used to generate the
107                              file.
108                         </xsd:documentation>
109                    </xsd:annotation>
110               </xsd:element>
111               <xsd:element name="schema_version" type="xsd:decimal"
112                    minOccurs="1" maxOccurs="1">
113                    <xsd:annotation>
114                         <xsd:documentation xml:lang="en">
115                              The required schema_version element
116                              specifies the version of the schema
117                              that the document has been written
118                              against and that should be used for
119                              validation.
120                         </xsd:documentation>
121                    </xsd:annotation>
122               </xsd:element>
123               <xsd:element name="timestamp" type="xsd:dateTime"
124                    minOccurs="1" maxOccurs="1">
125                    <xsd:annotation>
126                         <xsd:documentation xml:lang="en">
127                              The required timestamp element
128                              specifies when the particular document
129                              was compiled. The format for the
130                              timestamp is yyyy-mm-ddThh:mm:ss. Note
131                              that the timestamp element does not
132                              specify item in the document was
133                              created or modified but rather when
134                              the actual XML document that contains
135                              the items was created. For example, a
136                              document might pull a bunch of
137                              existing items together, each of which
138                              having been created at some point in
139                              the past. The timestamp in this case
140                              would be when this combined document
141                              was created.
142                         </xsd:documentation>
```

```
                    </xsd:annotation>
               </xsd:element>
               <xsd:any minOccurs="0" maxOccurs="unbounded"
                    namespace="##other" processContents="lax"/>
                </xsd:sequence>
          </xsd:complexType>
          <xsd:complexType name="ItemType">
               <xsd:annotation>
                    <xsd:documentation xml:lang="en">
                         The ItemType complex type defines an
                         element that represents a single CPE Name.
                         The required name attribute is a URI which
                         must be a unique key and should follow
                         the URI structure outlined in the CPE
                         Specification. The optional title element
                         is used to provide a human-readable title
                         for the platform. To support uses intended
                         for multiple languages, this element
                         supports the ‘xml:lang’ attribute. At most
                         one title element can appear for each
                         language. The notes element holds optional
                         descriptive material. Multiple notes
                         elements are allowed, but only one per
                         language should be used. Note that the
                         language associated with the notes element
                         applies to all child note elements. The
                         optional references element holds external
                         info references. The optional check element
                         is used to call out an OVAL Definition that
                         can confirm or reject an IT system as an
                         instance of the named platform. Additional
                         elements not part of the CPE namespace are
                         allowed and are just skipped by validation.
                         In essence, a dictionary file can contain
                         additional information the a user can
                         choose to use or not, but this information
                         is not required to be used or understood.
                    </xsd:documentation>
               </xsd:annotation>
               <xsd:sequence>
                    <xsd:element name="title"
                         type="cpe_dict:TextType" minOccurs="1"
                         maxOccurs="unbounded"/>
                    <xsd:element name="notes"
                         type="cpe_dict:NotesType" minOccurs="0"
                         maxOccurs="unbounded"/>
                    <xsd:element name="references"
                         type="cpe_dict:ReferencesType"
                         minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="check"
                         type="cpe_dict:CheckType" minOccurs="0"
                         maxOccurs="unbounded"/>
```

```
                    <xsd:any minOccurs="0" maxOccurs="unbounded"
                         namespace="##other" processContents="lax"/>
               </xsd:sequence>
               <xsd:attribute name="name" type="cpe_dict:namePattern"
                    use="required"/>
               <xsd:attribute name="deprecated" type="xsd:boolean"
                    use="optional" default="false"/>
               <xsd:attribute name="deprecated_by"
                    type="cpe_dict:namePattern" use="optional"/>
               <xsd:attribute name="deprecation_date"
                    type="xsd:dateTime" use="optional"/>
          </xsd:complexType>
          <xsd:complexType name="ListType">
               <xsd:annotation>
                    <xsd:documentation xml:lang="en">
                         The ListType complex type defines an
                         element that is used to hold a collection
                         of individual items. The required generator
                         section provides information about when the
                         definition file was compiled and under what
                         version. Additional elements not part of
                         the CPE namespace are allowed and are just
                         skipped by validation. In essence, a
                         dictionary file can contain additional
                         information the a user can choose to use or
                         not, but this information is not required
                         to be used or understood.
                    </xsd:documentation>
               </xsd:annotation>
               <xsd:sequence>
                    <xsd:element name="generator"
                         type="cpe_dict:GeneratorType" minOccurs="0"
                         maxOccurs="1"/>
                    <xsd:element ref="cpe_dict:cpe-item"
                         minOccurs="1" maxOccurs="unbounded"/>
                    <xsd:any minOccurs="0" maxOccurs="unbounded"
                         namespace="##other"
                           processContents="lax"/>
                </xsd:sequence>
          </xsd:complexType>
          <xsd:complexType name="TextType">
               <xsd:annotation>
                    <xsd:documentation xml:lang="en">
                         The TextType complex type allows the
                         xml:lang attribute to associate a specific
                         language with an element's string content.
                    </xsd:documentation>
               </xsd:annotation>
               <xsd:simpleContent>
                    <xsd:extension base="xsd:string">
                         <xsd:attribute ref="xml:lang"/>
                    </xsd:extension>
```

```
            </xsd:simpleContent>
        </xsd:complexType>
        <xsd:complexType name="NotesType">
            <xsd:annotation>
                <xsd:documentation xml:lang="en">
                    The notesType complex type defines an
                    element that consists of one or more child
                    note elements. It is assumed that each of
                    these note elements are representative of
                    the same language as defined by their
                    parent.
                </xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                <xsd:element name="note" type="xsd:string"
                    minOccurs="1" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute ref="xml:lang"/>
        </xsd:complexType>
       <xsd:complexType name="ReferencesType">
            <xsd:annotation>
                <xsd:documentation xml:lang="en">
                    The ReferencesType complex type defines an
                    element used to hold a collection of
                    individual references. Each reference
                    consists of a piece of text (intended to be
                    human-readable) and a URI (intended to be a
                    URL, and point to a real resource) and is
                    used to point to extra descriptive
                    material, for example a supplier's web site
                    or platform documentation.
                </xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                <xsd:element name="reference" minOccurs="1"
                    maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:simpleContent>
                            <xsd:extension
                                base="xsd:string">
                                <xsd:attribute name="href"
                                    type="xsd:anyURI"/>
                            </xsd:extension>
                        </xsd:simpleContent>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="CheckType">
            <xsd:annotation>
               <xsd:documentation xml:lang="en">
                  The CheckType complex
```

```
299                   type is used to define an element for hold
300                   information about an individual check. It
301                   includes a checking system specification URI,
302                   string content, and an optional external
303                  file reference. The checking system specification
304                   should be the URI for a particular version of
305                   OVAL or a related system testing language, and
306                   the content will be an identifier of a test
307                   written in that language. The external file
308                   reference could be used to point to the file in
309                   which the content test identifier is defined.
                   </xsd:documentation>
310             </xsd:annotation>
311             <xsd:simpleContent>
312                  <xsd:extension base="xsd:string">
313                       <xsd:attribute name="system"
314                            type="xsd:anyURI" use="required"/>
315                       <xsd:attribute name="href"
316                            type="xsd:anyURI" use="optional"/>
317                  </xsd:extension>
318             </xsd:simpleContent>
319        </xsd:complexType>
320  <!-- ======================================================== -->
321  <!-- ==================ID PATTERNS ======================== -->
322  <!-- ======================================================== -->
323        <xsd:simpleType name="namePattern">
324             <xsd:annotation>
325                  <xsd:documentation xml:lang="en">
326                       Define the format for acceptable CPE Names.
327                       A URN format is used with the id starting
328                       with the word cpe followed by :/ and
329                        then some number of individual components
330                       separated by colons.
331                  </xsd:documentation>
332             </xsd:annotation>
333             <xsd:restriction base="xsd:anyURI">
334                  <xsd:pattern value="[c][pP][eE]:/[AHOaho]?(:[A-
335                       Za-z0-9\._\-~%]*){0,6}"/>
336             </xsd:restriction>
337        </xsd:simpleType>
338  </xsd:schema>
```

1561 **Figure 10-8: CPE 2.2 Schema**

1562

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3     xmlns="http://scap.nist.gov/schema/cpe-extension/2.3"
4     xmlns:xml="http://www.w3.org/XML/1998/namespace"
5     targetNamespace="http://scap.nist.gov/schema/cpe-
6     extension/2.3" elementFormDefault="qualified"
7     attributeFormDefault="unqualified">
```

```
8
9     <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
10          schemaLocation="http://www.w3.org/2001/xml.xsd"/>
11
12    <xsd:element name="cpe23-item" type="itemType"/>
13
14    <xsd:complexType name="itemType">
15          <xsd:sequence>
16               <xsd:element name="provenance-record"
17                    type="provenanceRecordType" minOccurs="0"/>
18               <xsd:element name="deprecation" type="deprecationType"
19                    minOccurs="0" maxOccurs="unbounded"/>
20               <xsd:any minOccurs="0" maxOccurs="unbounded"
21                    namespace="##other" processContents="lax"/>
22          </xsd:sequence>
23          <xsd:attribute name="name" type="namePattern"
24               use="required"/>
25    </xsd:complexType>
26
27    <xsd:complexType name="deprecationType">
28          <xsd:sequence>
29               <xsd:element name="deprecated-by"
30                    type="deprecatedInfoType" minOccurs="1"
31                    maxOccurs="unbounded" form="qualified"/>
32          </xsd:sequence>
33          <xsd:attribute name="date" type="xsd:dateTime"
34               use="optional"/>
35    </xsd:complexType>
36
37    <xsd:complexType name="provenanceRecordType">
38          <xsd:sequence>
39               <xsd:element name="submitter" type="organizationType"
40                    minOccurs="1"/>
41               <xsd:element name="authority" type="organizationType"
42                    minOccurs="1" maxOccurs="unbounded"/>
43               <xsd:element name="change-description"
44                    type="changeDescriptionType" minOccurs="1"
45                    maxOccurs="unbounded"/>
46               <xsd:any minOccurs="0" maxOccurs="unbounded"
47                    namespace="##other" processContents="lax"/>
48          </xsd:sequence>
49    </xsd:complexType>
50
51    <xsd:complexType name="changeDescriptionType">
52          <xsd:sequence>
53               <xsd:element name="evidence-reference"
54                    type="evidenceReferenceType" minOccurs="0"/>
55               <xsd:element name="comments" type="xsd:token"
56                    minOccurs="0"/>
57          </xsd:sequence>
58          <xsd:attribute name="change-type" type="changeTypeType"
59               use="required"/>
```

```
60          <xsd:attribute name="date" type="xsd:dateTime"/>
61     </xsd:complexType>
62
63     <xsd:complexType name="evidenceReferenceType">
64          <xsd:simpleContent>
65               <xsd:extension base="xsd:anyURI">
66                    <xsd:attribute name="evidence"
67                         type="evidenceType" use="required"/>
68               </xsd:extension>
69          </xsd:simpleContent>
70     </xsd:complexType>
71
72     <xsd:complexType name="organizationType">
73          <xsd:sequence>
74          <xsd:element name="description" type="xsd:token"
75               minOccurs="0"/>
76          </xsd:sequence>
77          <xsd:attribute name="system-id" type="xsd:anyURI"
78               use="required"/>
79          <xsd:attribute name="name" type="xsd:token"
80               use="required"/>
81          <xsd:attribute name="date" type="xsd:dateTime"
82               use="required"/>
83     </xsd:complexType>
84
85     <xsd:complexType name="deprecatedInfoType">
86          <xsd:attribute name="name" type="searchableCpeName"
87          <xsd:attribute name="type" type="deprecationTypeType"
88          use="required">
89          <xsd:annotation>
90                    <xsd:documentation xml:lang="en">
91                         The type of deprecation associated with
92                         the deprecated-by element.  The type
93                         chosen will drive name resolution.
94                    </xsd:documentation>
95               </xsd:annotation>
96        </xsd:attribute>
97     </xsd:complexType>
98
99  <xsd:simpleType name="changeTypeType">
100         <xsd:restriction base="xsd:token">
101              <xsd:enumeration value="ORIGINAL_RECORD"/>
102              <xsd:enumeration value="DEPRECATION"/>
103              <xsd:enumeration value="DEPRECATION_MODIFICATION"/>
104         </xsd:restriction>
105    </xsd:simpleType>
106
107    <xsd:simpleType name="evidenceType">
108         <xsd:restriction base="xsd:token">
109              <xsd:enumeration value="CURATOR_UPDATE"/>
110              <xsd:enumeration value="VENDOR_FIX"/>
```

```
111             <xsd:enumeration value="THIRD_PARTY_FIX"/>
112       </xsd:restriction>
113  </xsd:simpleType>
114
115  <xsd:simpleType name="deprecationTypeType">
116       <xsd:restriction base="xsd:token">
117             <xsd:enumeration value="NAME_CORRECTION"/>
118             <xsd:enumeration value="NAME_REMOVAL"/>
119             <xsd:enumeration value="ADDITIONAL_INFORMATION"/>
120        </xsd:restriction>
121  </xsd:simpleType>
122
123      <!-- NOTE: Will need to change based on new CPE format -->
124  <xsd:simpleType name="namePattern">
125       <xsd:annotation>
126             <xsd:documentation xml:lang="en">
127                   Pattern defining identifier WFN (no embedded *,?)
128             </xsd:documentation>
129       </xsd:annotation>
130       <xsd:restriction base="xsd:anyURI">
131             <xsd:pattern value=""/>
132       </xsd:restriction>
133  </xsd:simpleType>
134
135      <!-- NOTE: Will need to change based on new CPE format -->
136  <xsd:simpleType name="searchableCpeName">
137       <xsd:annotation>
138             <xsd:documentation xml:lang="en">
139                   Pattern defining format of expression WFNs (can
140                   contain *, ?)
141             </xsd:documentation>
142       </xsd:annotation>
143        <xsd:restriction base="xsd:anyURI">
144             <xsd:pattern value=""/>
145       </xsd:restriction>
146  </xsd:simpleType>
147
148  </xsd:schema>
```

Figure 10-9: CPE 2.3 Extension of 2.2 Schema

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <cpe-list xmlns="http://cpe.mitre.org/dictionary/2.0"
3     xmlns:cpe23="http://scap.nist.gov/schema/cpe-extension/2.3"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5          xsi:schemaLocation="http://cpe.mitre.org/dictionary/2.0
6          http://cpe.mitre.org/files/cpe-dictionary_2.2.xsd
7               http://scap.nist.gov/schema/cpe-extension/2.3
8          cpe-extension_2.3.xsd">
9
10    <generator>
```

```
        <product_name>
             National Vulnerability Database (NVD)
        </product_name>
        <product_version>
             3.0-SNAPSHOT (DEVELOPMENT)
        </product_version>
        <schema_version>2.2</schema_version>
        <timestamp>2010-05-27T23:40:00</timestamp>
   </generator>

   <cpe-item name="cpe:/a:adobe:acrobat:3">
        <title xml:lang="en-US">Adobe Acrobat</title>
        <cpe23:cpe23-item
             name="cpe23:a:adobe:acrobat:3:*:*:*:*:*:*:*">
             <cpe23:provenance-record>
                  <cpe23:submitter system-
                       id="http://nvd.nist.gov" name="NVD"
                       date="2006-05-04T18:13:51.0Z"/>
                  <cpe23:authority system-
                       id="http://nvd.nist.gov" name="NVD"
                       date="2006-05-04T18:13:51.0Z"/>
                  <cpe23:change-description
                       change-type="ORIGINAL_RECORD"
                       date="2006-05-04T18:13:51.0Z">
                       <cpe23:evidence-reference
                            evidence="CURATOR_UPDATE">
                            http:/adobe.com/versionHistory
                       </cpe23:evidence-reference>
                       <cpe23:comments>
                            Any comments relating to the
                            evidence of why this was updated
                            should go here. Also the
                            description of the change should
                            be explained.
                       </cpe23:comments>
                  </cpe23:change-description>
                  <cpe23:change-description
                       change-type="DEPRECATION"
                       date="2007-05-04T18:13:51.0Z">
                       <cpe23:evidence-reference
                            evidence="CURATOR_UPDATE">
                            http:/adobe.com/versionHistory
                       </cpe23:evidence-reference>
                       <cpe23:comments>
                            This name was deprecated
                       </cpe23:comments>
                  </cpe23:change-description>
             </cpe23:provenance-record>
             <cpe23:deprecation date="2006-05-04T18:13:51.0Z">
                  <cpe23:deprecated-by name="cpe
                       23:a:adobe:acrobat:3.0:*:*:*:*:*:*:*"
                       type="NAME_CORRECTION"/>
```

```
63              </cpe23:deprecation>
64         </cpe23:cpe23-item>
65    </cpe-item>
66
67
68    <cpe-item name="cpe:/a:adobe:acrobat:3.0">
69         <title xml:lang="en-US">Adobe Acrobat 3.0</title>
70         <cpe23:cpe23-item
71              name="cpe23:a:adobe:acrobat:3.0:*:*:*:*:*:*:*">
72              <cpe23:provenance-record>
73                   <cpe23:submitter system-id="http://nvd.nist.gov"
74                        name="NVD" date="2007-05-04T18:13:51.0Z"/>
75                   <cpe23:authority system-id="http://nvd.nist.gov"
76                        name="NVD" date="2007-05-04T18:13:51.0Z"/>
77                   <cpe23:change-description
78                        change-type="ORIGINAL_RECORD"
79                        date="2007-05-04T18:13:51.0Z">
80                        <cpe23:evidence-reference
81                             evidence="CURATOR_UPDATE">
82                             http:/adobe.com/versionHistory
83                        </cpe23:evidence-reference>
84                        <cpe23:comments>
85                             deprecated previous with better
86                             name
87                        </cpe23:comments>
88                   </cpe23:change-description>
89              </cpe23:provenance-record>
90         </cpe23:cpe23-item>
91    </cpe-item>
92 </cpe-list>
```

Figure 10-10: Sample CPE 2.3 Dictionary Instance Data

Appendix D—Change Log

Release 0 – 9 June 2010

- Complete draft specification released to the CPE community for comment.

Release 1 – 30 June 2010

- Minor edits to audience description.
- Minor editorial changes throughout the document.
- Removed all mention of and support for the logical value UNKNOWN.
- Updated Dictionary Searching section to remove the notion of an Error result, and clarify on superset versus subset matches.
- Updated deprecation logic, and data model to include three distinct types of deprecation.